

Two-tape finite automata with quantum and classical states

Shenggen Zheng^{a,*}, Lvzhou Li^{a,†}, Daowen Qiu^{a,b,c,‡}

^a*Department of Computer Science, Sun Yat-sen University, Guangzhou 510006, China*

^b*SQIG–Instituto de Telecomunicações, IST, TULisbon,
Av. Rovisco Pais 1049-001, Lisbon, Portugal*

^c*The State Key Laboratory of Computer Science, Institute of Software,
Chinese Academy of Sciences, Beijing 100080, China*

Abstract

Two-way finite automata with quantum and classical states (2QCFA) were introduced by Ambainis and Watrous, and *two-way two-tape deterministic finite automata* (2TFA) were introduced by Rabin and Scott. In this paper we study 2TFA and propose a new computing model called *two-way two-tape finite automata with quantum and classical states* (2TQCFA). First, we give efficient 2TFA algorithms for recognizing languages which can be recognized by 2QCFA. Second, we give efficient 2TQCFA algorithms to recognize several languages whose status vis-a-vis 2QCFA have been posed as open questions, such as $L_{\text{square}} = \{a^n b^{n^2} \mid n \in \mathbf{N}\}$. Third, we show that $\{a^n b^{n^k} \mid n \in \mathbf{N}\}$ can be recognized by $(k+1)$ -tape deterministic finite automata ($(k+1)$ TFA). Finally, we introduce k -tape automata with quantum and classical states (k TQCFA) and prove that $\{a^n b^{n^k} \mid n \in \mathbf{N}\}$ can be recognized by k TQCFA.

Keywords: Quantum computing models; Quantum finite automata;

1 Introduction

Interest in quantum computation has steadily increased since Shor’s quantum algorithm for factoring integers in polynomial time [21] and Grover’s algorithm of searching in database of size n with only $O(\sqrt{n})$ accesses [14]. As we know, these algorithms are based on *quantum Turing machines* which are complicated to implement using today’s experiment technology. Therefore, it is natural to consider much more restricted quantum computing models.

*E-mail address: zhengshenggen@gmail.com

†E-mail address: lilvzhou@gmail.com

‡Corresponding author. E-mail address: issqdw@mail.sysu.edu.cn (D. Qiu).

Classically, as one of the simplest computing models, *deterministic finite automata* (DFA) and *nondeterministic finite automata* (NFA) have been deeply studied [15]. Furthermore, *two-way two-tape deterministic finite automata* (2TFA) and *multi-tape finite automata* (m TFA) were first introduced by Rabin and Scott in a seminal paper in 1959 [22]. Unlike DFA or NFA, 2TFA can recognize not only *regular languages* (RL) but also some *context-free languages* (CFL), and even some *no-context-free languages* (NCFL). In this paper, we will show that some *context-free languages*, and even some *no-context-free languages* can be recognized by 2TFA in linear time.

Correspondingly, it may be interesting to consider restricted *quantum Turing machines*, such as *quantum finite automata* (QFA). QFA were first introduced independently by Kondacs and Watrous [10], as well as Moore and Crutchfield [11]. As a quantum variant of FA, QFA have attracted wide attentions in the academic community [2, 3, 4, 5, 6, 7, 8, 9]. Many kinds of QFA have been proposed and studied(e.g., see [13]). QFA are mainly divided into two kinds: *one-way quantum finite automata* (1QFA) and *two-way quantum finite automata* (2QFA). If we compare 1QFA with their classical counterparts DFA, 1QFA have some weaknesses since they can recognize only a proper subset of RL [3]. 2QFA, however, are more powerful than their classical counterparts *two-way deterministic finite automata* (2DFA) which recognize only RL. 2QFA were first introduced by Kondacs and Watrous, and then they proved that $L_{eq} = \{a^n b^n \mid n \in \mathbf{N}\}$ and $L_{trieq} = \{a^n b^n c^n \mid n \in \mathbf{N}\}$ can be recognized by 2QFA [10], where the first language is CFL, and the second one is NCFL. However, 2QFA have a disadvantage in the sense that we need at least $\mathcal{O}(\log n)$ qubits to store the position of the tape head, which is relative to the length of the input. In order to conquer this disadvantage, Ambainis and Watrous proposed *two-way finite automata with quantum and classical states* (2QCFA)[1], an intermediate model between 1QFA and 2QFA, which are still more powerful than their classical counterparts 2DFA. A 2QCFA is essentially a classical 2DFA augmented with a quantum component of constant size, where the dimension of the associated Hilbert space does not depend on the length of the input. Ambainis and Watrous showed $L_{pal} = \{\omega \in \{a, b\}^* \mid \omega = \omega^R\}$ and $L_{eq} = \{a^n b^n \mid n \in \mathbf{N}\}$ can be recognized by 2QCFA [1]. Several other no-regular languages including $M_{eq} = \{a^n b_1^n a^m b_2^m \mid n, m \in \mathbf{N}\}$, $L_{eq}(k, a) = \{a^{kn} b^n \mid n \in \mathbf{N}\}$ and $L_+ = \{\omega \in \{a, b\}^* \mid \#\omega(a) = \#\omega(b)\}$ were proved to be recognized by 2QCFA in Qiu's paper [12].

In this paper we revisit *two-tape two-way deterministic finite automata* (2TFA), of which the definition has a little difference from the original model introduced by Rabin and Scott in 1959 [22]. When we mention 2TFA in the following, we refer to the new defined model without ambiguity. First, we prove that all the languages mentioned above that can be recognized by 2QFA or 2QCFA, including L_{pal} , L_{eq} , L_{trieq} , M_{eq} , $L_{eq}(k, a)$ and L_+ , can also be recognized by 2TFA in linear time. Note that 2QCFA need polynomial or exponential time to recognize these languages. Second, We show some languages, including $L_{copy} = \{\omega\omega \mid \omega \in \Sigma^*\}$, $L_{middle} = \{xay \mid x, y \in \Sigma^*, a \in \Sigma\}$, and $L_{balanced} = \{x \in \{"(, "\)"\}^* \mid \text{parentheses in } x \text{ are balanced}\}$

balanced }, can be recognized by 2TFA in linear time. However, as far as we know, it is still not clear whether 2QFA or 2QCFA can recognize these languages or not. Thus, it seems that 2TFA are more powerful than 2QFA or 2QCFA.

Based on 2TFA, we propose 2TQCFA which are similar to 2QCFA, only augmented with another tape. 2TQCFA can be implemented with a quantum part of constant size, and they are shown to be powerful. We consider the language: $L_{\text{square}} = \{a^n b^{n^2} \mid n \in \mathbf{N}\}$, whether it can be recognized by 2QFCA or not is still an open problem proposed by Ambainis and Watrous in [1]. Rao proposed a new model of automata called *Two-Way Optical Interference Automata* (2OIA) which use the phenomenon of interference, and proved that they can recognize L_{square} [20]. However, 2OIA are complicated to implement, because it is hard to control the phenomenon of interference in high precision. In this paper, we show that L_{square} can be recognized by 2TQCFA in polynomial time.

Also, we consider *multi-tape deterministic finite automata* (m TFA). We show that $L_{k\text{pow}} = \{a^n b^{n^k} \mid n \in \mathbf{N}\}$ can be recognized by $(k+1)$ -tape deterministic finite automata $((k+1)\text{TFA})$ in linear time. Furthermore, we propose a new model of quantum automata, *multi-tape finite automata with quantum and classical states* (m TQCFA), which is essentially the model of m TFA augmented with a quantum component of constant size. We prove that $L_{k\text{pow}}$ can be recognized by k -tape finite automata with quantum and classical states (k TQCFA) in polynomial time.

The remainder of this paper has the following organization. In Section 2 we give the definitions of 2TFA, 2TQCFA, m TFA and m TQCFA. Then, in Section 3 we show 2TFA can recognize those languages mentioned before. Afterwards, in Section 4 we describe a 2TQCFA for recognizing L_{square} . Subsequently, in Section 5 we describe a $(k+1)$ TFA and a k TQCFA for recognizing $L_{k\text{pow}}$. Finally we make a conclusion in Section 6.

2 Definitions

2.1 Definition of 2TFA

The definition here is slightly different from the original model introduced by Rabin and Scott [22]. A 2TFA is defined by a 6-tuple

$$\mathcal{M} = (S, \Sigma, \delta, s_0, S_{\text{acc}}, S_{\text{rej}}) \quad (1)$$

where,

- S is a finite set of classical states;
- Σ is a finite set of input symbols; the tape symbol set is $\Gamma = \Sigma \cup \{\emptyset, \$\}$; two tapes with

the same input ω are $\mathcal{T}_1 = \mathcal{C}_1\omega\$_1$ and $\mathcal{T}_2 = \mathcal{C}_2\omega\$_2$; we assume that the input in \mathcal{T}_i is started by the left end-marker \mathcal{C}_i and terminated by the right end-marker $\$_i$;

- δ is the transition function:

$$S \setminus (S_{acc} \cup S_{rej}) \times \Gamma \rightarrow S \times D_1 \times D_2 \quad (2)$$

where $D_1 = D_2 = \{\leftarrow, \downarrow, \rightarrow\}$, of which the element in turn means left, stationary and right, respectively; we restrict that at least one of D_1 and D_2 is stationary in this mapping, which means that a 2TFA can at most move one head and scan one new symbol at one time. If both of the tape heads are stationary, the symbol to be scanned is the same as the previous step. We assume that the first symbol scanned is \mathcal{C}_1 . So we can simplify the transition function to take “ Γ ” as variable instead of “ $\Gamma \times \Gamma$ ”, which is the main difference from the original model introduced by Rabin and Scott. The mapping can be interpreted as follows: $\delta(s, \gamma) = (s', d_1, d_2)$ means a machine currently in state $s \in S \setminus (S_{acc} \cup S_{rej})$, scanning symbol $\gamma \in \Gamma$ will change state to s' , and the tape head i will move according to the directions d_1 and d_2 ;

- $s_0 \in S$ is the initial state of the machine;
- $S_{acc} \subset S$ and $S_{rej} \subset S$ are the sets of accepting states and rejecting states, respectively.

For an input string $\omega = \sigma_1 \dots \sigma_l \in \Sigma^*$, the content of the first tape is $\mathcal{T}_1 = \mathcal{C}_1\sigma_1 \dots \sigma_l\$_1$, and the content of the second tape is $\mathcal{T}_2 = \mathcal{C}_2\sigma_1 \dots \sigma_l\$_2$. The computation of a 2TFA on input string ω is the sequence $(p_0, i_0^1, i_0^2), \dots, (p_j, i_j^1, i_j^2), \dots$, in which $p_j \in S$, $0 \leq i_j^1, i_j^2 \leq l+1$, and

- The initial configuration $(p_0, i_0^1, i_0^2) = (s_0, 0, 0)$;
- If the instance configuration is (p_j, i_j^1, i_j^2) and the transition function is $\delta(p_j, \sigma) = (p_{j+1}, d_j^1, d_j^2)$, then the next configuration will be $(p_{j+1}, i_j^1 + d_j^1, i_j^2 + d_j^2)$.

A computation is assumed to halt if and only if an accepting state or a rejecting state is reached. If there is a configuration (p_j, i_j^1, i_j^2) with $p_j \in S_{acc}$, then the automaton is said to accept ω ; else if $p_j \in S_{rej}$, then the automaton is said to reject ω .

For related basic notations and further details about *finite automata*, the reader may refer to [15, 19, 23].

2.2 Definition of 2TQCFA

2QCFA were introduced by Ambainis and Watrous in [1]. The new model that we define here is similar to 2QCFA, but with two tapes. Informally, we describe a 2TQCFA as a 2TFA which has access to a constant size of quantum register, upon which it performs quantum

transformations and measurements. Before giving a more formal definition of 2TQCFA, we review a few basic facts regarding quantum computing. We would refer the reader to [16, 17, 18] for a more detailed overview of quantum computing. Let $\mathcal{H}(Q)$ represents the Hilbert space with the corresponding base identified with set Q . If $\{|i_Q\rangle\}$ is an orthonormal base for $\mathcal{H}(Q)$, then state $q \in \mathcal{H}(Q)$ can be written in the form $\sum_i^n a_i |i_Q\rangle$, where a_i is a complex number and $\sum_i^n |a_i|^2 = 1$. Let $\mathcal{U}(\mathcal{H}(Q))$ and $\mathcal{M}(\mathcal{H}(Q))$ denote the sets of unitary operators and orthogonal measurements over $\mathcal{H}(Q)$, respectively. A unitary operator $U \in \mathcal{U}(\mathcal{H}(Q))$ is an invertible linear operator which preserves length. A set of *projective measurements* $\mathcal{M} = \{P_i\} \in \mathcal{M}(\mathcal{H}(Q))$ are *Hermitian operators* on $\mathcal{H}(Q)$ such that $P_i = P_i^\dagger$, $P_i^2 = P_i^\dagger$ for all i ; $P_i P_j = 0$ for $i \neq j$, and $\sum_i P_i = I$. Upon measuring the state $|\psi\rangle$, the probability of getting result i is given by $p(i) = \langle\psi|P_i|\psi\rangle$. Given that the result i occurred, the state of the quantum system immediately changes to $P_i|\psi\rangle/\sqrt{\langle\psi|P_i|\psi\rangle}$.

A 2TQCFA is specified by a 9-tuple

$$\mathcal{M} = (Q, S, \Sigma, \Theta, \delta, q_0, s_0, S_{acc}, S_{rej}) \quad (3)$$

where,

- Q is a finite set of quantum states;
- S is a finite set of classical states;
- Σ is a finite set of input symbols; the tape symbol set is $\Gamma = \Sigma \cup \{\$\}$; two tapes with the same input ω are $\mathcal{T}_1 = \$\omega\$$ and $\mathcal{T}_2 = \$\omega\$$; we assume that the input in \mathcal{T}_i is started by the left end-marker $\$$ and terminated by the right end-marker $\$$;
- Θ is the transition function of quantum states:

$$S \setminus (S_{acc} \cup S_{rej}) \times \Gamma \rightarrow \mathcal{U}(\mathcal{H}(Q)) \cup \mathcal{M}(\mathcal{H}(Q)), \quad (4)$$

where each action $\Theta(s, \gamma)$ corresponds to either a unitary transformation or a projective measurement;

- δ is the transition function of classical states. If $\Theta(s, \gamma) \in \mathcal{U}(\mathcal{H}(Q))$, then δ is:

$$S \setminus (S_{acc} \cup S_{rej}) \times \Gamma \rightarrow S \times D_1 \times D_2, \quad (5)$$

which is the same transition function as we have defined in 2TFA. If $\Theta(s, \gamma) \in \mathcal{M}(\mathcal{H}(Q))$, and the resulting set of the measurement is $R = \{r_1, r_2, \dots, r_n\}$, then δ is:

$$S \setminus (S_{acc} \cup S_{rej}) \times \Gamma \times R \rightarrow S \times D_1 \times D_2, \quad (6)$$

where $\delta(s, \gamma)(r_i) = (s', d_1, d_2)$ means that when the projective measurement result is r_i with probability p_i , then $s \in S$ scanning $\gamma \in \Gamma$ will be changed to state s' , and the next scanned symbol will be decided by d_1 and d_2 ;

- $q_0 \in Q$ is the initial quantum state;
- $s_0 \in S$ is the initial classical state;
- $S_{acc} \subset S$ and $S_{rej} \subset S$ are the sets of classical accepting states and rejecting states, respectively.

Given an input ω , a 2TQCFA $\mathcal{M} = (Q, S, \Sigma, \Theta, \delta, q_0, s_0, S_{acc}, S_{rej})$ operates as follows:

At the beginning, tape head i is at \mathbb{C}_i , the quantum initial state is $|q_0\rangle$, the classical initial state is s_0 , and $|q_0\rangle$ will be changed according to $\Theta(s_0, \mathbb{C}_1)$.

- If $\Theta(s_0, \mathbb{C}_1) = U \in \mathcal{U}(\mathcal{H}(Q))$, then the quantum state evolves as $|q_0\rangle \rightarrow U|q_0\rangle$, and meanwhile, the classical state s_0 will be changed to s' according to $\delta(s_0, \mathbb{C}_1) = (s', d_1, d_2)$. The symbol scanned next is in \mathcal{T}_1 , if $d_1 \neq \downarrow$; the symbol scanned next is in \mathcal{T}_2 , if $d_2 \neq \downarrow$; otherwise, it holds that $d_1 = d_2 = \downarrow$, and then the symbol scanned next is unchanged.
- If $\Theta(s_0, \mathbb{C}_1) = M \in \mathcal{M}(\mathcal{H}(Q))$, $M = \{P_1, \dots, P_m\}$, then $|q_0\rangle \rightarrow \frac{P_i|q_0\rangle}{\sqrt{\langle q_0|P_i|q_0\rangle}}$ with result r_i . Meanwhile, we have $\delta(s_0, \mathbb{C}_1)(r_i) = (s_i, d_1, d_2)$. If $s_i \in S_{acc}$, \mathcal{M} accepts ω ; if $s_i \in S_{rej}$, \mathcal{M} rejects ω ; otherwise, \mathcal{M} scans the next symbol, and the transformation is similar to the above process.

A computation is assumed to halt if and only if an accepting state or a rejecting classical state is reached. Let $p_{acc}(\omega)$ and $p_{rej}(\omega)$ denote the accepting and rejecting probabilities for \mathcal{M} with input ω , respectively. We say that a language L is recognized by a 2TQCFA \mathcal{M} if L is recognized by a 2TQCFA \mathcal{M} with one-sided error ϵ , which means that for all $\omega \in \Sigma^*$, $p_{acc}(\omega) + p_{rej}(\omega) = 1$, $p_{acc}(\omega) = 1$ if $\omega \in L$, and $p_{rej}(\omega) \geq 1 - \epsilon$ if $\omega \notin L$.

2.3 Definitions of m TFA and m TQCFA

The definition of k TFA is similar to that of 2TFA. A k TFA is a 6-tuple

$$M = (S, \Sigma, \delta, s_0, S_{acc}, S_{rej}) \tag{7}$$

where S, Σ, s_0, S_{acc} and S_{rej} are just like the ones in 2TFA. The map of δ is: $S \setminus (S_{acc} \cup S_{rej}) \times \Gamma \rightarrow S \times D^k$. We restrict that at most one of $D_i (i = 1, 2, \dots, k)$ is no stationary in the mapping and all the tapes have the same input ω . The transformation is similar to that of 2TFA.

An m TQCFA is essentially a classical m TFA augmented with a quantum component of constant size.

3 Languages recognized by 2TFA

When we use a 2DFA to recognize a language, we can not record how many specific symbols have been scanned. For example, when we use a 2DFA to recognize $L_{eq} = \{a^n b^n \mid n \in \mathbf{N}\}$, we can not record how many symbols a have been scanned, which is the reason that 2DFA can not recognize L_{eq} . Even if a 2TFA is used, we are still not able to record how many specific symbols have been scanned. But we have two tapes, so we can compare the numbers of the specific symbols having been scanned between the two tapes, which leads to that 2TFA can recognize more languages and become more powerful.

3.1 Languages recognized by 2QFA or 2QCFA that can also be recognized by 2TFA

L_{eq} and L_{trieq} were proved to be recognized by 2QFA in [10], L_{pal} and L_{eq} were proved to be recognized by 2QCFA in [1], and M_{eq} , $L_{eq}(k, a)$ and L_+ were proved to be recognized by 2QCFA in [12]. In this subsection we will prove all these languages can be recognized by 2TFA in linear time. We assume that $\mathbf{N} = \{0, 1, 2, \dots, n, \dots\}$, and we use $\#\omega(\sigma)$ and $|\omega|$ to denote the number of symbol σ in string ω and the length of string ω , respectively.

Theorem 1. $L_{eq} = \{a^n b^n \mid n \in \mathbf{N}\}$ can be recognized by 2TFA in linear time.

Proof. The idea of this proof is as follows: First we check whether the input ω is of the form $a^+ b^+$, and then we scan \mathcal{T}_1 from right to left to consume symbol b and scan \mathcal{T}_2 from left to right to consume symbol a alternately. If we scan a symbol b in \mathcal{T}_1 , the next symbol scanned in \mathcal{T}_2 must be symbol a ; otherwise, we reject the input ω . Once we scan a symbol a in \mathcal{T}_1 , which means that the tape head of \mathcal{T}_1 reaches the a^+ section of \mathcal{T}_1 , we turn to \mathcal{T}_2 to see which symbol it scans. If the next symbol scanned in \mathcal{T}_2 is symbol b , we accept the input ω ; otherwise, we reject it. A 2TFA \mathcal{M} for L_{eq} is defined as follows:

$$\mathcal{M} = (S, \Sigma, \delta, s_0, S_{acc}, S_{rej}) \quad (8)$$

where,

- $S = \{s_0, s_1, s_2, s_a, s_b, s_{crt}, s_{acc}, s_{rej}\}$, where s_0, s_1 and s_2 are used to check whether ω is of the form $a^+ b^+$; s_a is used to consume symbol a in \mathcal{T}_2 and s_b is used to consume symbol b in \mathcal{T}_1 ; s_{crt} is the critical state which will be changed to s_{acc} or s_{rej} depending on the next scanned symbol;
- $\Sigma = \{a, b\}$; s_0 is the initial state; $S_{acc} = \{s_{acc}\}$; $S_{rej} = \{s_{rej}\}$;
- The mapping δ is defined as follows:

$$\begin{array}{lll}
\delta(s_0, \emptyset) = (s_0, \rightarrow, \downarrow) & \delta(s_0, a) = (s_1, \rightarrow, \downarrow) \\
\delta(s_0, b) = (s_{rej}, -, -) & \delta(s_0, \$) = (s_{acc}, -, -) \\
\delta(s_1, a) = (s_1, \rightarrow, \downarrow) & \delta(s_1, b) = (s_2, \rightarrow, \downarrow) & \delta(s_1, \$) = (s_{rej}, -, -) \\
\delta(s_2, a) = (s_{rej}, -, -) & \delta(s_2, b) = (s_2, \rightarrow, \downarrow) & \delta(s_2, \$) = (s_b, \leftarrow, \downarrow) \\
\delta(s_b, a) = (s_{crt}, \downarrow, \rightarrow) & \delta(s_b, b) = (s_a, \downarrow, \rightarrow) \\
\delta(s_a, a) = (s_b, \leftarrow, \downarrow) & \delta(s_a, b) = (s_{rej}, -, -) \\
\delta(s_{crt}, a) = (s_{rej}, -, -) & \delta(s_{crt}, b) = (s_{acc}, -, -)
\end{array}$$

Remark :When \mathcal{M} comes to halting states, the directions of the tape heads will be meaningless, and thus we use “-”.

It is easy to verify that L_{eq} can be recognized by \mathcal{M} . Checking whether the input ω is of the form a^+b^+ takes $|\omega|$ time. Scanning \mathcal{T}_1 from right to left and scanning \mathcal{T}_2 from left to right alternately takes $2|\omega|$ time at the worst case. Hence, the worst running time of \mathcal{M} is $\mathbf{O}(|\omega|)$, which is linear time.

□

Theorem 2. $L_{pal} = \{\omega \in \{a, b\}^* \mid \omega = \omega^R\}$ can be recognized by 2TFA in linear time.

Proof. We scan \mathcal{T}_1 from right to left and scan \mathcal{T}_2 from left to right alternately, and each time we make sure that the symbols scanned are the same. If the symbols scanned are not the same, we reject the input ω ; otherwise, when the tape head of \mathcal{T}_1 reaches the left end-marker and the tape head of \mathcal{T}_2 reaches the right end-marker, we accept it. A 2TFA \mathcal{M} for L_{pal} is defined as follows:

$$\mathcal{M} = (S, \Sigma, \delta, s_0, S_{acc}, S_{rej}) \quad (9)$$

where,

- $S = \{s_0, s_t, s, s_a, s_b, s_{crt}, s_{acc}, s_{rej}\}$, where s_0 and s_t are used to move the first tape head to the right end-marker of \mathcal{T}_1 ; s_a and s_b are used to consume symbol a and b in \mathcal{T}_2 , respectively; s is used to consume a symbol in \mathcal{T}_1 ; s_{crt} is the critical state which will be changed to s_{acc} or s_{rej} depending on the next scanned symbol;
- $\Sigma = \{a, b\}$; s_0 is the initial state; $S_{acc} = \{s_{acc}\}$; $S_{rej} = \{s_{rej}\}$.
- The mapping δ is defined as follows:

$$\begin{array}{lll}
\delta(s_0, \sigma) = (s_0, \rightarrow, \downarrow) \text{ for } \sigma = a, b, \emptyset & \delta(s_0, \$) = (s_t, \leftarrow, \downarrow) \\
\delta(s_t, \sigma) = (s_\sigma, \downarrow, \rightarrow) \text{ for } \sigma = a, b & \delta(s_t, \emptyset) = (s_{acc}, -, -) \\
\delta(s_\sigma, \sigma') = (s, \leftarrow, \downarrow) \text{ for } \sigma = \sigma' & \delta(s_\sigma, \sigma') = (s_{rej}, -, -) \text{ for } \sigma \neq \sigma' \\
\delta(s, \sigma) = (s_\sigma, \downarrow, \rightarrow) \text{ for } \sigma = a, b & \delta(s, \emptyset) = (s_{crt}, \downarrow, \rightarrow) \\
\delta(s_{crt}, \sigma) = (s_{rej}, -, -) \text{ for } \sigma \neq \$ & \delta(s_{crt}, \$) = (s_{acc}, -, -)
\end{array}$$

It is easy to verify that L_{pal} can be recognized by \mathcal{M} . It takes \mathcal{M} exactly $3|\omega|$ time to accept the input ω and less than $3|\omega|$ time to reject it. So L_{pal} can be recognized by 2TFA in linear time. \square

Theorem 3. $L_{trieq} = \{a^n b^n c^n \mid n \in \mathbf{N}\}$ can be recognized by 2TFA in linear time.

Proof. The 2TFA \mathcal{M} to recognize this language is similar to the one recognizing L_{eq} . The idea of this proof is as follows: First, we check whether the input ω is of the form $a^+b^+c^+$; second, we use two subroutines of 2TFA \mathcal{M}' which recognizes L_{eq} to check whether the number of symbol c is equal to the number of symbol a and b . We scan \mathcal{T}_1 from right to left and \mathcal{T}_2 from left to right alternately to make sure the number of symbol c in \mathcal{T}_1 is equal to the number of symbol a in \mathcal{T}_2 . We scan \mathcal{T}_1 from left to right and \mathcal{T}_2 from left to right alternately to make sure the number of symbol c in \mathcal{T}_1 is equal to the number of symbol b in \mathcal{T}_2 . The details of 2TFA \mathcal{M} for L_{trieq} are omitted here.

It is easy to verify that L_{trieq} can be recognized by \mathcal{M} . It takes \mathcal{M} $3|\omega|$ time to accept the ipnput ω and less than $3|\omega|$ to reject it at the worst cases. So L_{trieq} can be recognized by 2TFA in linear time. \square

Theorem 4. $L_{eq}(k, a) = \{a^{kn} b^n \mid n \in \mathbf{N}\}$ can be recognized by 2TFA in linear time.

Proof. The 2TFA \mathcal{M} to recognize this language is similar to the one recognizing L_{eq} . First we check whether the input ω is of the form a^+b^+ . But, we scan \mathcal{T}_1 from right to left to consume one symbol b and scan \mathcal{T}_2 from left to right to consume k symbols a instead of just one symbol alternately. A 2TFA \mathcal{M} for $L_{eq}(k, a)$ is defined as follows:

$$\mathcal{M} = (S, \Sigma, \delta, s_0, S_{acc}, S_{rej}) \quad (10)$$

where,

- $S = \{s_0, s_1, s_2, s_{a1}, s_{a2}, \dots, s_{ak}, s_b, s_{crt}, s_{acc}, s_{rej}\}$, where s_0, s_1 and s_2 are used to check whether ω is of the form a^+b^+ ; s_{ai} is used to consume symbol a in \mathcal{T}_2 and s_b is used to consume symbol b in \mathcal{T}_1 ; s_{crt} is the critical state which will be changed to s_{acc} or s_{rej} depending on the next scanned symbol;
- $\Sigma = \{a, b\}$; s_0 is the initial state; $S_{acc} = \{s_{acc}\}$; $S_{rej} = \{s_{rej}\}$;
- The mapping δ is defined as follows:

$$\begin{array}{lll}
\delta(s_0, \emptyset) = (s_0, \rightarrow, \downarrow) & \delta(s_0, a) = (s_1, \rightarrow, \downarrow) \\
\delta(s_0, b) = (s_{rej}, -, -) & \delta(s_0, \$) = (s_{acc}, -, -) \\
\delta(s_1, a) = (s_1, \rightarrow, \downarrow) & \delta(s_1, b) = (s_2, \rightarrow, \downarrow) & \delta(s_1, \$) = (s_{rej}, -, -) \\
\delta(s_2, a) = (s_{rej}, -, -) & \delta(s_2, b) = (s_2, \rightarrow, \downarrow) & \delta(s_2, \$) = (s_b, \leftarrow, \downarrow) \\
\delta(s_b, a) = (s_{crt}, \downarrow, \rightarrow,) & \delta(s_b, b) = (s_{a1}, \downarrow, \rightarrow,) \\
\delta(s_{ai}, a) = (s_{a(i+1)}, \downarrow, \rightarrow) & \text{for } i = 1, 2, \dots, k-1 \\
\delta(s_{ai}, b) = (s_{rej}, -, -) & \text{for } i = 1, 2, \dots, k-1 \\
\delta(s_{ak}, a) = (s_b, \leftarrow, \downarrow) & \delta(s_{ak}, b) = (s_{rej}, -, -) \\
\delta(s_{crt}, a) = (s_{rej}, -, -) & \delta(s_{crt}, b) = (s_{acc}, -, -)
\end{array}$$

It takes $\mathcal{M} 3|\omega|$ time at the worst cases. So $L_{eq}(k, a)$ can be recognized by 2TFA in linear time. \square

Theorem 5. $M_{eq} = \{a^n b_1^n a^m b_2^m \mid n, m \in \mathbf{N}\}$ can be recognized by 2TFA in linear time.

Proof. The 2TFA \mathcal{M} to recognize this language is similar to the one recognizing L_{trieq} . We check whether the input ω is of the form $a^+ b_1^+ a^+ b_2^+$ at the beginning. Next we use two subroutines of 2TFA \mathcal{M}' which recognizes L_{eq} . First, we scan \mathcal{T}_1 from left to right and \mathcal{T}_2 from left to right alternately to make sure the number of symbol a in the first section of \mathcal{T}_1 is equal to the number of symbol b_1 in \mathcal{T}_2 . Second, we scan \mathcal{T}_1 from left to right and \mathcal{T}_2 from left to right alternately to make sure the number of symbol a in the second section of \mathcal{T}_1 is equal to the number of symbol b_2 in \mathcal{T}_2 . We do not bore the details of 2TFA \mathcal{M} for M_{eq} here.

It is easy to verify that M_{eq} can be recognized by \mathcal{M} . It takes $\mathcal{M} 2|\omega|$ time at the worst cases. So M_{eq} can be recognized by 2TFA in linear time. \square

Theorem 6. $L_+ = \{\omega \in \{a, b\}^* \mid \#\omega(a) = \#\omega(b)\}$ can be recognized by 2TFA in linear time.

Proof. The 2TFA \mathcal{M} to recognize this language is similar to the one recognizing L_{eq} . But we do not need to check whether ω is of the form $a^+ b^+$. We scan \mathcal{T}_1 for symbol a from left to right and scan \mathcal{T}_2 for symbol b from left to right alternately. We start at looking for symbol a in \mathcal{T}_1 . If the symbol scanned is not symbol a , we skip it and look for the next symbol in \mathcal{T}_1 until we scan symbol a . Once we have scanned a symbol a in \mathcal{T}_1 , we look for a symbol b in \mathcal{T}_2 . If the symbol scanned is not symbol b , we just skip it, and look for the next symbol in \mathcal{T}_2 until we scan symbol b or come to the right end-marker. If the head of \mathcal{T}_2 reaches the right end-marker, we reject the input ω ; otherwise, turn to \mathcal{T}_1 . When the tape head of \mathcal{T}_1 reaches the right end-marker, if there is no more symbol b waiting for being scanned in \mathcal{T}_2 , we accept the input ω ; otherwise, we reject it. A 2TFA \mathcal{M} for L_{eq} is defined as follows:

$$M = (S, \Sigma, \delta, s_0, S_{acc}, S_{rej}) \tag{11}$$

where,

- $S = \{s_0, s_a, s_b, s_{crt}, s_g, s_l, s_{eq}\}$, where s_a is used to consume symbol a in \mathcal{T}_1 and s_b is used to consume symbol b in \mathcal{T}_2 ; s_{crt} is the critical state to check whether ω is in L ; s_g means $\#\omega(a) > \#\omega(b)$, s_l means $\#\omega(a) < \#\omega(b)$, and s_{eq} means $\#\omega(a) = \#\omega(b)$, respectively;
- $\Sigma = \{a, b\}$; s_0 is the initial state; $S_{acc} = \{s_{eq}\}$; $S_{rej} = \{s_g, s_l\}$;
- The mapping δ is defined as follows:

$$\begin{array}{lll}
\delta(s_0, \mathbb{C}_1) = (s_0, \rightarrow, \downarrow) & \delta(s_0, \$1) = (s_{acc}, -, -) \\
\delta(s_0, a) = (s_b, \downarrow, \rightarrow,) & \delta(s_0, b) = (s_0, \downarrow, \rightarrow,) \\
\delta(s_b, a) = (s_b, \downarrow, \rightarrow) & \delta(s_b, b) = (s_a, \rightarrow, \downarrow) & \delta(s_b, \$2) = (s_g, -, -) \\
\delta(s_a, a) = (s_b, \downarrow, \rightarrow) & \delta(s_a, b) = (s_a, \rightarrow, \downarrow) & \delta(s_a, \$1) = (s_{crt}, \downarrow, \rightarrow) \\
\delta(s_{crt}, a) = (s_{crt}, \downarrow, \rightarrow) & \delta(s_{crt}, b) = (s_l, -, -) & \delta(s_{crt}, \$2) = (s_{eq}, -, -)
\end{array}$$

It is easy to verify that $L_=$ can be recognized by \mathcal{M} . It takes $\mathcal{M} 2|\omega|$ time to accept the input ω and no more than $2|\omega|$ time to reject it. So $L_=$ can be recognized by 2TFA in linear time. \square

Having proved several languages that can be recognized by 2TFA, next we will prove a more general language $L_{linear} = \{\sigma_1^{k_1} \sigma_2^{k_2} \dots \sigma_m^{k_m} \mid a_i k_i = \sum_{j=1, j \neq i}^m a_j k_j, \Sigma = \{\sigma_1, \sigma_2, \dots, \sigma_m\}, m, k_1, k_2, \dots, k_m, a_i \in \mathbf{N}; a_1, a_2, \dots, a_{i-1}, a_{i+1}, \dots, a_m \in \{0, 1\}\}$ can be recognized by 2TFA in linear time. L_{eq} , L_{trieq} , M_{eq} and $L_{eq}(k, a)$ are just the special cases of this language. L_{linear} can also be recognized by 2QCFA in polynomial time.

Theorem 7. $L_{linear} = \{\sigma_1^{k_1} \sigma_2^{k_2} \dots \sigma_m^{k_m} \mid a_i k_i = \sum_{j=1, j \neq i}^m a_j k_j, \Sigma = \{\sigma_1, \dots, \sigma_m\}, m, k_1, k_2, \dots, k_m, a_i \in \mathbf{N}; a_1, a_2, \dots, a_{i-1}, a_{i+1}, \dots, a_m \in \{0, 1\}\}$ can be recognized by 2TFA in linear time.

Proof. $\omega \in L_{linear}$ has at most $m - 1$ restrictions such as $a_i k_i = \sum_{j=1, j \neq i}^m a_j k_j$ (if it has more than $m - 1$ restrictions, they must be relative, and we can replace them by less than $m - 1$ restrictions), where $m = |\Sigma|$ which is a constant. First, we use \mathcal{T}_1 to check whether ω is of the form $\sigma_1^* \sigma_2^* \dots \sigma_m^*$, and then we move the tape head of \mathcal{T}_1 back to the left end-maker. This takes $2|\omega|$ time. The 2TFA \mathcal{M} for L_{linear} consists of $m - 1$ subroutines, and each subroutine verifies one restriction. 2TFA \mathcal{M} accepts ω if all restrictions are satisfied. As an example, for a restriction $a_i k_i = \sum_{j=1, j \neq i}^m a_j k_j$, symbol σ_i can be regarded as symbol a , $\sigma_j (j \neq i, a_j \neq 0)$ can be regarded as symbol b , and $\sigma_j (j \neq i, a_j = 0)$ can be regarded as symbol c . This subroutine is equivalent to the subroutine of recognizing $\{\omega \in \{a, b, c\}^* \mid a_i \#\omega(a) = \#\omega(b)\}$, which is easy to implement. We scan \mathcal{T}_1 from left to right to consume one symbol a and scan \mathcal{T}_2 from left to right to consume a_i symbols b , and we just skip symbol c . This subroutine takes $2|\omega|$ time. After a subroutine ends, we move both tape heads to the left end-markers, which takes

$2|\omega|$ time. So L_{linear} can be recognized by a 2TFA with running time $(4(m - 1) + 2)|\omega|$ at the worst cases, which is linear time.

□

Remark : L_{linear} can be recognized by 2QCFA. We can used m-1 subroutines to verify the m-1 restrictions, and the idea is similar to the proof of the above theorem.

3.2 Languages recognized by 2TFA but whether or not they can be recognized by 2QFA or 2QCFA is still pending

In this subsection we will show that several languages can be recognized by 2TFA in linear time, including L_{copy} , L_{middle} and $L_{balanced}$. Whether L_{middle} and $L_{balanced}$ can be recognized by 2QCFA or 2QFA or not still remains as an open problem [1].

Theorem 8. $L_{copy} = \{\omega\omega \mid \omega \in \Sigma^*\}$ can be recognized by 2TFA in linear time.

Proof. First, we check whether the length of the input is even. If it is odd, the input is rejected; otherwise, we continue to do the next steps. The key to solve this problem is to find the middle of the input. In terms of the advantage of two tapes, we can easily find the middle as follows: Tape \mathcal{T}_1 scans two symbols from left to right while tape \mathcal{T}_2 scans one symbol from left to right. When tape \mathcal{T}_1 reaches the right end-marker, tape \mathcal{T}_2 will just go to the middle exactly. After doing this, we move the tape head of \mathcal{T}_1 to the left end-marker C_1 , and then compare the symbols between \mathcal{T}_1 and \mathcal{T}_2 one by one from left to right until tape \mathcal{T}_2 reaches the end. If all the symbols compared are the same, the input is accepted; otherwise, it is rejected. Obviously, this can be done in linear time. □

Theorem 9. $L_{middle} = \{xay \mid x, y \in \Sigma^*, a \in \Sigma\}$ can be recognized by 2TFA in linear time.

Proof. First, we check whether the length of the input is odd. If it is even, the input is rejected; otherwise, we continue to do the following steps. The key to solve this problem is still to find the middle of the input. We can find it easily in the same way as we did in the previous theorem. We check whether the symbol in the middle is a . If the symbol is a , the input is accepted; otherwise, it is rejected. Obviously, this can be done in linear time. □

Theorem 10. $L_{balanced} = \{x \in \{"(", ")"\}^* \mid \text{parentheses in } x \text{ are balanced}\}$ can be recognized by 2TFA in linear time.

Proof. In order to prove this theorem, we need to prove a lemma. We use $F(w)$ to denote the set of all the prefixes of w except empty string λ and w .

Lemma 11. A string w is in $L_{balanced}$ if and only if w satisfies

- (1). $\#_w("(") = \#_w(")")$ and
- (2). $\#_x("(") \geq \#_x(")")$ for all $x \in F(w)$.

Proof. 1 : First, we show that if w satisfies $\#_w("(") = \#_w(")")$ and $\#_x("(") \geq \#_x(")")$ for all $x \in F(w)$, then $w \in L_{balanced}$. We prove this by induction on the length of w .

BASIS: $|w| = 0$ is obvious. When $|w| = 2$, let $w = ab$, where $a, b \in \{"(", ")"\}$. We have $F(w) = \{a\}$ according to the definition. Because $\#_a("(") \geq \#_a(")")$, a must be $"("$. Because $\#_w("(") = \#_w(")")$, b must be $")"$. Therefore $w = "()"$, which follows that $w \in L_{balanced}$.

INDUCTION: Suppose that when $|w| \leq 2n(n \geq 1)$, the result holds. We will prove it holds for $|w| = 2(n+1)$.

Case1 There exists a string $u \in F(w)$ which satisfies $\#_u("(") = \#_u(")")$. In this case, let $w = uv(u, v \in \{"(", ")"\}^*)$. Obviously, u satisfies condition(1) and (2), and meanwhile we have $|u| \leq 2n$. According to the assumption, we get $u \in L_{balanced}$. Furthermore, we have $\#_v("(") = \#_w("(") - \#_u("(") = \#_w(")") - \#_u(")") = \#_v(")"$). So we get v satisfies condition(1). For every $x \in F(v)$, we have $\#_x("(") = \#_{ux}("(") - \#_u("(") \geq \#_{ux}(")") - \#_u(")") = \#_x(")")$. So we get v satisfies condition (2). Because $|v| \leq 2n$, we get $v \in L_{balanced}$ according to the assumption. It follows that $w \in L_{balanced}$.

Case2 Otherwise, for every $u \in F(w)$ satisfies $\#_u("(") > \#_u(")")$. Let $w = aw'b$. According to condition(2), we have $\#_a("(") \geq \#_a(")")$, and thus we get $a = "("$. Because $|aw'|$ is odd, we have $\#_{aw'}("(") > \#_{aw'}(")")$ according to condition (2). According to condition(1), we have $\#_w("(") = \#_w(")")$, and thus it holds that $b = ")"$. It follows that $\#_{w'}("(") = \#_w("(") - 1 = \#_w(")") - 1 = \#_{w'}(")"$). Thus w' satisfies condition(1). For every $v \in F(w')$, we have $\#_v("(") = \#_{av}("(") - 1 > \#_{av}(")") - 1 \geq \#_{av}(")") = \#_v(")"$), thus w' satisfies condition(2). Meanwhile, we have $|w'| = 2n$, and thus we get $w' \in L_{balanced}$ according to the assumption. It follows that $w \in L_{balanced}$.

2 : We prove the other direction here. If $w \in L_{balanced}$, then w satisfies condition (1) and (2). We prove this by induction on the length of w .

BASIS: $|w| = 0$ is obvious. When $|w| = 2$, it must be the case $w = "()"$. Obviously, it satisfies condition (1) and (2).

INDUCTION: Suppose that when $|w| \leq 2n(n \geq 1)$, the result holds. We will prove it holds for $|w| = 2(n+1)$.

Case1 w can be divided into two parts as uv , where $u, v \in L_{balanced}$. Obviously, we have $|u| \leq 2n$ and $|v| \leq 2n$. According to the assumption, we get u and v

satisfy condition (1) and (2). It follows that $\#_w(") = \#_u(") + \#_v(") = \#_u(") + \#_v(") = \#_w(")$, which says w satisfies condition (1). For every $x \in F(w)$, if $|x| \leq |u|$, then we have $x \in F(u)$ and $\#_x(") \geq \#_x("$; otherwise, let $x = uy$, and then we have $y \in F(v)$. Therefore, it follows that $\#_x(") = \#_u(") + \#_y(") \geq \#_u(") + \#_y(") = \#_x("$). So we get w satisfies condition (2).

Case2 w can not be divided into two parts as uv such that $u, v \in L_{balanced}$. In this case, w must be of the form $("w'")$, where $w' \in L_{balanced}$ and $|w'| = 2n$. According to the assumption, we have $w' \in L_{balanced}$ satisfies condition (1) and (2). It follows that $\#_w(") = \#_{w'}(") + 1 = \#_{w'}(") + 1 = \#_w("$, which says w satisfies condition(1). For every $x \in F(w)$, let $x = "y$. Then we have $y \in F(w')$. Therefore we have $\#_x(") = 1 + \#_y(") \geq 1 + \#_y(") > \#_y(") = \#_x("$), i.e., w satisfies condition(2).

Therefore, the lemma has been proved. \square

Now we return to the proof of the theorem. According to the lemma proved above, it is enough for us to check whether the input w satisfies condition (1) and (2). We use tape \mathcal{T}_1 to scan the input w from left to right, and use tape \mathcal{T}_2 to save the result $R = \#_x(") - \#_x("$) for every $x \in F(w)$. If the tape head of \mathcal{T}_2 is on \emptyset_2 , it means that $R = 0$; if the tape head of \mathcal{T}_2 is on the i th square of w , it means that $R = i$. When tape \mathcal{T}_1 scans a symbol $"$, the tape head of \mathcal{T}_2 moves right by one square; when tape \mathcal{T}_1 scans a symbol $"$, the tape head of \mathcal{T}_2 moves left by one square. If the tape head of \mathcal{T}_2 is on \emptyset_2 , and the symbol scanned by the tape head of \mathcal{T}_1 is $"$, then the input is rejected. When the tape head of \mathcal{T}_1 comes to the right end-marker of w , and the tape head of \mathcal{T}_2 is on \emptyset_2 , then the input is accepted. Obviously, this can be done in linear time. Therefore, we have proved the theorem. \square

4 Languages recognized by 2TQCFA

Ambainis and Watrous introduced a model of quantum automata called 2QCFA and proposed an open problem of whether $L_{square} = \{a^n b^{n^2} \mid n \in \mathbf{N}\}$ can be recognized by 2QCFA or not [1]. Although this problem remains open for 2QCFA, we will propose a new model of quantum automata called 2TQCFA by which L_{square} can be recognized in polynomial time. In addition, Ambainis and Watrous did not give the details on how to use a 2QCFA to simulate a coin flip that is an essential component for 2QCFA to recognize languages. Thus we will give the details here.

Theorem 12. *For any $\epsilon > 0$, there is a 2TQCFA \mathcal{M} that accepts any $\omega \in L_{square} = \{a^n b^{n^2} \mid n \in \mathbf{N}\}$ with certainly, rejects any $\omega \notin L_{square} = \{a^n b^{n^2} \mid n \in \mathbf{N}\}$ with probability at least $1 - \epsilon$ and halts in polynomial time.*

Proof. The main idea is as follows: First, we verify whether it holds that $\omega \in \{a^n b^{kn} \mid n \in \mathbf{N}\}$. This can be done in linear time, and the proof is similar to the proof of Theorem 4. Second, we verify whether it holds that $k = n$. We consider a 2TQCFA \mathcal{M} with 2 quantum states $|q_0\rangle$ and $|q_1\rangle$. \mathcal{M} starts with the quantum state $|q_0\rangle$. Firstly, we scan \mathcal{T}_1 , and every time when \mathcal{M} scans symbol a in \mathcal{T}_1 , the quantum state rotated by angle $\alpha = \sqrt{2}\pi$. After all symbols a in \mathcal{T}_1 have been scanned, we scan symbol b in \mathcal{T}_1 and symbol a in \mathcal{T}_2 alternately. When the number of symbol b in \mathcal{T}_1 having been scanned equals to the total number of symbol a in \mathcal{T}_2 , the quantum state is rotated by $-\alpha$. When the end of \mathcal{T}_1 is reached, we measure the quantum state. If the measurement result is $|q_1\rangle$, ω is rejected. Otherwise, the process is repeated.

After having verified $\omega \in \{a^n b^{kn} \mid n \in \mathbf{N}\}$, the quantum state is rotated by α when \mathcal{M} scans symbol a in \mathcal{T}_1 , and the quantum state is rotated by $-\alpha$ when \mathcal{M} has scanned n symbols b in \mathcal{T}_1 in the process presented above. If $k = n$, rotations of quantum state cancel one another and the final quantum state will be $|q_0\rangle$ with certainty. Otherwise, the final state will be a superposition state because $\sqrt{2}$ is irrational, and the amplitude of $|q_1\rangle$ in the state is sufficiently large, which means that repeating the process $\mathbf{O}(n^2)$ times guarantees getting $|q_1\rangle$ at least once with high probability. \mathcal{M} also needs to halt and accept input $\omega \in \{a^n b^{n^2} \mid n \in \mathbf{N}\}$ after repeating the process $\mathbf{O}(n^2)$ times rather than repeating it forever. To achieve this, We periodically execute a subroutine that accepts with a small probability c/n^2 , which is much smaller than the probability of getting $|q_1\rangle$. If $\omega \notin \{a^n b^{n^2} \mid n \in \mathbf{N}\}$, this does not have much influence.

The process is described as follows:

1 If the input $\omega = \lambda$, accept.

Check whether ω is of the form $a^n b^{kn}$ ($n > 0, k > 0$). If not, reject.

2 Otherwise, repeat the following routine infinitum:

(1) Move the tape head of \mathcal{T}_1 to \mathbb{C}_1 and move the tape head of \mathcal{T}_2 to \mathbb{C}_2 . Set the quantum state to be $|q_0\rangle$.

(2) While the currently scanned symbol is not $\$_1$, do the following:

1) If the currently scanned symbol of \mathcal{T}_1 is a

Perform U_α on the quantum state and move the tape head of \mathcal{T}_1 one square right.

2) Else if the currently scanned symbol of \mathcal{T}_1 is b

Turn to \mathcal{T}_2 . Move the tape head of \mathcal{T}_2 one square right and scan a symbol.

a) If the scanned symbol of \mathcal{T}_2 is a

Turn to \mathcal{T}_1 , move the tape head of \mathcal{T}_1 one square right and scan a symbol.

b) Else if the scanned symbol of \mathcal{T}_2 is b

 Perform $U_{-\alpha}$ on the quantum state. Move the tape head of \mathcal{T}_2 back to \mathbb{C}_2 .

 Turn to \mathcal{T}_1 , move the tape head of \mathcal{T}_1 one square right and scan a symbol.

(3) Perform $U_{-\alpha}$ on the quantum state.

(4) Measure the quantum state. If the result is $|q_1\rangle$, reject.

(5) Repeat the following subroutine two times:

 1) Move the tape head of \mathcal{T}_1 back to \mathbb{C}_1 , and move the tape head of \mathcal{T}_2 back to \mathbb{C}_2 .

 2) Move the tape head of \mathcal{T}_1 one square right and scan a symbol.

 3) While the currently scanned symbol is not \mathbb{C}_1 or $\$_1$, do the following:

 Simulate a coin flip.

A . If the result is “head”:

 Move the tape head of \mathcal{T}_1 one square right and scan a symbol.

 a) If the symbol scanned of \mathcal{T}_1 is a , do nothing.

 b) If the symbol scanned of \mathcal{T}_1 is b

 Turn to \mathcal{T}_2 , move the tape head of \mathcal{T}_2 one square right and scan a symbol.

 ◊ If the scanned symbol of \mathcal{T}_2 is a

 Turn to \mathcal{T}_1 , move the tape head of \mathcal{T}_1 one square right, scan a symbol and goto b).

 ◊ Else if the scanned symbol of \mathcal{T}_2 is b

 Move the tape head of \mathcal{T}_2 to \mathbb{C}_2 . Turn to \mathcal{T}_1 , and move the tape head of \mathcal{T}_1 one square left.

B . Else

 Move the tape head of \mathcal{T}_1 left and scan a symbol.

 a) If the symbol scanned of \mathcal{T}_1 is a , do nothing.

 b) If the symbol scanned of \mathcal{T}_1 is b

 Turn to \mathcal{T}_2 , move the tape head of \mathcal{T}_2 one square right and scan a symbol.

 ◊ If the scanned symbol of \mathcal{T}_2 is a

 Turn to \mathcal{T}_1 , move the tape head of \mathcal{T}_1 one square left and scan a symbol. Goto b).

 c) Move the tape head of \mathcal{T}_2 to \mathbb{C}_2 . Turn to \mathcal{T}_1 , and move the tape head of \mathcal{T}_1 one square right.

(6) If both times the process ends at the right end-marker of \mathcal{T}_1 , simulate l coin flips.

 If all the results are ”heads”, accept.

A 2TQCFA \mathcal{M} to recognize L_{square} is defined as follows:

$$\mathcal{M} = (Q, S, \Sigma, \Theta, \delta, q_0, s_0, S_{acc}, S_{rej}) \quad (12)$$

where,

- $Q = \{q_0, q_1\}$;
- S is a finite set of classical states;
- $\Sigma = \{a, b\}$; $\mathcal{C}_1, \mathcal{C}_2$ are the left end-markers of \mathcal{T}_1 and \mathcal{T}_2 , respectively; $\mathbb{S}_1, \mathbb{S}_2$ are the right end-markers of \mathcal{T}_1 and \mathcal{T}_2 , respectively;
- q_0 is the initial quantum state;
- s_0 is the initial classical state;
- $S_{acc} = \{s_{acc}\}$;
- $S_{rej} = \{s_{rej}\}$.

In fact, the key step is to construct quantum and classical transition functions Θ and δ . We construct them according to the process described above.

For **1**: Decide whether the input ω is of the form $a^n b^{kn}$ ($n > 0, k > 0$). For the trivial case $a^n b^{kn}$ ($n = 0$) we accept it at this subroutine. In this subroutine, we do not change the quantum state. The quantum transition function is the identity I . The classical transition function is similar to the one recognizing $w \in \{a^{kn} b^n \mid n \in \mathbf{N}\}$, and we omit the details here.

For **2-(1)**: Move the tape heads of \mathcal{T}_1 and \mathcal{T}_2 to \mathcal{C}_1 and \mathcal{C}_2 , respectively. In this subroutine, we do not change the quantum state. The quantum and classical transition functions are easy to describe. We omit the details here.

For **2-(2)** and **2-(3)**: Do the quantum operations.

- the classical states are $S_2 = \{s_{02}, s_{a2}, s_{b1}, s_{re3}, s_{crt2}, s_m\}$, where s_{02} is the starting state of this subroutine; s_{re3} is the state for moving back to \mathcal{C}_2 of \mathcal{T}_2 ; s_{a2} and s_{b1} are states to consume a symbol a and b from left to right in \mathcal{T}_2 and \mathcal{T}_1 , respectively; s_m is the ending state of this subroutine waiting to make a measurement for the quantum state.
- The quantum transition function is defined as follows:

$$\begin{aligned}\Theta(s_{02}, a) &= U_\alpha & \Theta(s_{a2}, b) &= U_{-\alpha} \\ \Theta(s_{crt2}, b) &= U_{-\alpha} \\ \Theta(s, \sigma) &= I \text{ for } s \in \{s_{b1}, s_{re3}\}\end{aligned}$$

- the classical transition function is defined as follows:

$$\begin{aligned}
\delta(s_{02}, a) &= (s_{02}, \rightarrow, \downarrow) & \delta(s_{02}, b) &= (s_{a2}, \downarrow, \rightarrow) \\
\delta(s_{a2}, a) &= (s_{b1}, \rightarrow, \downarrow) & \delta(s_{a2}, b) &= (s_{re3}, \downarrow, \leftarrow) \\
\delta(s_{re3}, a) &= (s_{re3}, \downarrow, \leftarrow) & \delta(s_{re3}, \mathbb{C}_2) &= (s_{a2}, \downarrow, \rightarrow) \\
\delta(s_{b1}, b) &= (s_{a2}, \downarrow, \rightarrow) & \delta(s_{b1}, \$1) &= (s_{crt2}, \downarrow, \rightarrow) \\
\delta(s_{crt2}, b) &= (s_m, \downarrow, \leftarrow)
\end{aligned}$$

For **2-(4)**: Measure the quantum state.

- the classical states are $S_3 = \{s_m, s_{re4}\}$.
- The quantum transition function is defined as follows:
 $\Theta(s_m, \sigma) = M = \{P_0, P_1\}$ for any $\sigma \in \Sigma$, where

$$P_0 = |0\rangle\langle 0|, P_1 = |1\rangle\langle 1|. \quad (13)$$

- the classical transition function is defined as follows:
 $\delta(s_m, \sigma)(1) = (s_{rej}, -, -), \quad \delta(s_m, \sigma)(0) = (s_{re4}, \downarrow, \leftarrow) \quad \text{for any } \sigma \in \Sigma.$

For **2-(5)**: Two times random walk. The random walk here is a little different from the random walk in Ambainis and Watrous's paper [1]. We still consider scanning a symbol a as one step, but we consider scanning n symbols b as one step in this random walk.

We do not bore all the details of quantum and classical transition functions here. In the random walk, every time we go on a step by scanning a symbol a or n symbols b . Simulation of a coin flip is an essential component in this subroutine. The machine \mathcal{M} simulates a coin flip according to the following transition functions, with $s_{03}, |q_0\rangle$ as the starting classical and quantum states, respectively.

Let projective measurement $M = \{P_0, P_1\}$, where

$$P_0 = |0\rangle\langle 0|, P_1 = |1\rangle\langle 1|. \quad (14)$$

The results 0 and 1 represent the results of coin flip "head" and "tail", respectively. Unitary operator U is defined as follow:

$$U = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}.$$

This operator changes the base state $|q_0\rangle$ or $|q_1\rangle$ to be a superposition state $|\psi\rangle$ or $|\phi\rangle$, respectively, as follows:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|q_0\rangle + |q_1\rangle), \quad (15)$$

$$|\phi\rangle = \frac{1}{\sqrt{2}}(|q_0\rangle - |q_1\rangle). \quad (16)$$

When measuring $|\psi\rangle$ or $|\phi\rangle$ with M , we will get the result 0 or 1 with probability of $\frac{1}{2}$, respectively. This is similar to the coin flip process. If the result is 0, we simulate a right step; if the result is 1, we simulate a left step.

$$\begin{aligned}\Theta(s_{03}, \sigma) &= U & \delta(s_{03}, \sigma) &= (s'_{03}, \downarrow, \downarrow) \\ \Theta(s'_{03}, \sigma) &= M \\ \delta(s'_{03}, \sigma)(0) &= (s_{03}, \rightarrow, \downarrow) & \delta(s'_{03}, \sigma)(1) &= (s_{03}, \leftarrow, \downarrow)\end{aligned}$$

For **2-(6)**: Simulate l coin flips. If all the results are “heads”, the input is accepted. The classic state starts from s_{cf} and the quantum state starts from $|q_1\rangle$. We still use unitary operator U and projective measurement M described as above to simulate a coin flip. We use the measurement result 0 and 1 to represent “head” and “tail”, respectively.

- the classical states are $\{s_{fail}, s_{acc}, s_{cf}^{(0,i)}, s_{cf}^{(1,i)}, i = 0, 1, 2, \dots, l+1\}$, where $s_{cf}^{(0,0)}$ is the starting state.
- The quantum and classical transition functions are defined as follows:
for $0 \leq i \leq l$,

$$\begin{aligned}\Theta(s_{cf}^{(0,i)}, \$1) &= U & \delta(s_{cf}^{(0,i)}, \$1) &= (s_{cf}^{(1,i)}, \downarrow, \downarrow) \\ \Theta(s_{cf}^{(1,i)}, \$1) &= M \\ \delta(s_{cf}^{(1,i)}, \$1)(1) &= (s_{fail}, \leftarrow, \downarrow) & \delta(s_{cf}^{(1,i)}, \$1)(0) &= (s_{cf}^{(0,i+1)}, \downarrow, \downarrow)\end{aligned}$$

When the classical state changes to s_{fail} , it means that the “tail” result of a coin flip has occurred. There is no need to simulate any more coin flips. The tape head of \mathcal{T}_1 should be moved back to the first symbol of the tape and another new iteration should be started. We omit the transition function here. When the classical state changes to $s_{cf}^{(0,l+1)}$, it means that all the results of the k coin flips are “heads”. The classical state should be changed to an accepting state. The transition function is as follow:

$$\delta(s_{cf}^{(0,l+1)}, \$1) = (s_{acc}, -, -).$$

Lemma 13. *If the input is $\omega = a^n b^k n$ and $k \neq n$, \mathcal{M} rejects after **2-(2),2-(3),2-(4)** with probability at least $1/2(n-k)^2$.*

Proof. The state $|q_0\rangle$ is rotated by $\sqrt{2}\pi$ when \mathcal{T}_1 scans one symbol a , and it is rotated by $-\sqrt{2}\pi$ when \mathcal{T}_1 scans n symbols b . The $|q_0\rangle$ is rotated by $\sqrt{2}(n-k)\pi$ after **2-(2),2-(3)**. So the quantum state of \mathcal{M} after rotating is $\cos(\sqrt{2}(n-k)\pi)|q_0\rangle + \sin(\sqrt{2}(n-k)\pi)|q_1\rangle$. In **2-(4)** the probability of observing $|q_1\rangle$ is $\sin^2(\sqrt{2}(n-k)\pi)$. Without loss of generality, we assume $n-k > 0$. Let l be the closest integer to $\sqrt{2}(n-k)$. Assume that $\sqrt{2}(n-k) > l$ (the other

case is symmetric), then $2(n-k)^2 > l^2$. So we get $2(n-k)^2 - 1 \geq l^2$ and $l \leq \sqrt{2(n-k)^2 - 1}$. We have

$$\sqrt{2}(n-k) - l \geq \sqrt{2}(n-k) - \sqrt{2(n-k)^2 - 1} \quad (17)$$

$$= \frac{(\sqrt{2}(n-k) - \sqrt{2(n-k)^2 - 1})(\sqrt{2}(n-k) + \sqrt{2(n-k)^2 - 1})}{\sqrt{2}(n-k) + \sqrt{2(n-k)^2 - 1}} \quad (18)$$

$$= \frac{1}{\sqrt{2}(n-k) + \sqrt{2(n-k)^2 - 1}} > \frac{1}{2\sqrt{2}(n-k)}. \quad (19)$$

Because l is the closest integer to $\sqrt{2}(n-k)$, we have $0 < \sqrt{2}(n-k) - l < 1/2$. Assume $f(x) = \sin(x\pi) - 2x$. We have $f''(x) = -\pi^2 \sin(x\pi) \leq 0$ when $x \in [0, 1/2]$. That is to say, $f(x)$ is concave in $[0, 1/2]$, and we have $f(0) = f(1/2) = 0$. So for any $x \in [0, 1/2]$, it holds that $f(x) \geq 0$, that is, $\sin(x\pi) \geq 2x$. Therefore

$$\sin^2(\sqrt{2}(n-k)\pi) = \sin^2((\sqrt{2}(n-k) - l)\pi) \quad (20)$$

$$\geq (2(\sqrt{2}(n-k) - l))^2 = 4(\sqrt{2}(n-k) - l)^2 \quad (21)$$

$$\geq 4\left(\frac{1}{2\sqrt{2}(n-k)}\right)^2 = \frac{1}{2(n-k)^2}. \quad (22)$$

So the lemma has been proved. \square

Lemma 14. *If the input is $\omega = a^n b^{kn}$ and $k \neq n$, \mathcal{M} accepts after 2-(5),2-(6) with probability $1/2^l(n+k+1)^2$.*

Proof. 2-(5) is similar to two times of random walk (takes scanning n symbols b as one step) starting at location 1 and ending at location 0 (the left end-marker \mathbb{C}_1) or at location $(k+1)n+1$ (the right end-marker $\$$). It can be known from probability theory that the probability of reaching the location $(k+1)n+1$ is $1/(n+k+1)$ (see Chapter14.2 in [24]). Repeating it twice and flipping l coins, we get the probability $1/2^l(n+k+1)^2$. \square

Let $l = 1 + \lceil \log \varepsilon \rceil$. If $\omega = a^n b^m, m \neq kn$, then step 1 always rejects it. If $\omega = a^n b^{n^2}$, 2-(2),2-(3),2-(4) always turn $|q_0\rangle$ to $|q_0\rangle$, and \mathcal{M} never rejects. After 2-(5),2-(6), the probability of \mathcal{M} accepting ω is $1/2^l(n+k+1)^2$. Repeating step 2 for cn^2 times, the accepting probability is $1 - (1 - \frac{1}{2^l(n+k+1)^2})^{cn^2}$, and this can be made arbitrarily close to 1 by selecting constant c appropriately. If $x = a^n b^{kn}$ and $k \neq n$, \mathcal{M} rejects after 2-(2),2-(3),2-(4) with probability $P_r > 1/2(n-k)^2$. \mathcal{M} accepts after 2-(5),2-(6) with probability $P_a = 1/2^l(n+k+1)^2 \leq \varepsilon/2(n+k+1)^2$. If we repeat step 2 indefinitely, the probability of rejecting is

$$\sum_{i \geq 0} (1 - P_a)^i (1 - P_r)^i P_r = \frac{P_r}{P_a + P_r - P_a P_r} \quad (23)$$

$$> \frac{P_r}{P_a + P_r} > \frac{1/2}{1/2 + \varepsilon/2} = \frac{1}{1 + \varepsilon} > 1 - \varepsilon. \quad (24)$$

If we assume the input $\omega = a^n b^m$, step **1** takes $\mathbf{O}(n + m)$ time, step **2-1** takes $\mathbf{O}(n + m)$ time, **2-2,2-3,2-4** take $\mathbf{O}(n + m)$ time, and **2-5,2-6** take $\mathbf{O}(n + m)^2$ time. The expected number of repeating step **2** is $\mathbf{O}((n + m/n)^2)$. Hence, the expected running time of \mathcal{M} is at most $\mathbf{O}((n + m/n)^2(n + m)^2)$. \square

5 Languages recognized by $m\text{TFA}$ and $m\text{TQCFA}$

We know that 2QCFA are more powerful than 2DFA, but we do not know whether $k\text{TQCFA}$ are more powerful than $k\text{TFA}$ or not. Although we can not prove $k\text{TQCFA}$ are strictly more powerful than $k\text{TFA}$, we will give an example which seems to imply that the result may be true. We will show that $\{a^n b^{n^k} \mid n \in \mathbf{N}\} (k = 1, 2, \dots)$ can be recognized by $(k + 1)\text{TFA}$ in linear time. Also, we will prove that $\{a^n b^{n^k} \mid n \in \mathbf{N}\} (k = 1, 2, \dots)$ can be recognized by $k\text{TQCFA}$ with one side-error in polynomial time. Thus, it seems that $k\text{TQCFA}$ are more powerful than $k\text{TFA}$.

Theorem 15. $L_{\text{square}} = \{a^n b^{n^2} \mid n \in \mathbf{N}\}$ can be recognized by 3TFA in linear time.

Proof. The process to recognize L_{square} is described as follows:

- 1** If the input $\omega = \lambda$, accept.
- 2** Check whether the input ω is of the form $a^+ b^+$. If not, reject.
- 3** Move the tape head of \mathcal{T}_1 to the first symbol b in \mathcal{T}_1 .
- 4** While the currently scanned symbol by \mathcal{T}_1 is b , do the following:
 - Turn to \mathcal{T}_2 , move the tape head of \mathcal{T}_2 one square right and scan a symbol.
 - (1)** If the currently scanned symbol by \mathcal{T}_2 is b
 - Move the tape head of \mathcal{T}_2 back to \mathcal{C}_2 . Turn to \mathcal{T}_3 , move the tape head of \mathcal{T}_3 one square right and scan a symbol.
 - (A)** If the currently scanned symbol by \mathcal{T}_3 is b
 - Reject.
 - (B)** Else
 - Turn to \mathcal{T}_1 , move the tape head of \mathcal{T}_1 one square right and scan a symbol.
 - (2)** Else
 - Turn to \mathcal{T}_1 , move the tape head of \mathcal{T}_1 one square right and scan a symbol.
 - 5** Turn to \mathcal{T}_2 , move the tape head of \mathcal{T}_2 one square right and scan a symbol.

- (1) If the currently scanned symbol by \mathcal{T}_2 is not b
Reject.
- (2) Else
Turn to \mathcal{T}_3 , move the tape head of \mathcal{T}_3 one square right and scan a symbol.
 - (A) If he currently scanned symbol by \mathcal{T}_2 is not b
Reject.
 - (B) Else, accept.

We assume the input $\omega = a^n b^m$. Let k_2 be the number of symbol a having been scanned by tape head \mathcal{T}_2 and k_3 be the number of symbol a having been scanned by tape head \mathcal{T}_3 from left to right. Actually, $k_2 k_3$ is acting like a 2 digits of n -radix number. When the tape head of \mathcal{T}_1 scans a symbol b , we just add 1 to k_2 . When k_2 is large enough ($k_2 = n$), we turn k_2 to 0 and add 1 to k_3 . It is easy to verify that this process will accept $L_{\text{square}} = \{a^n b^{n^2} \mid n \in \mathbf{N}\}$ in linear time.

□

The above result is easy to extend to the case of $(k+1)$ TFA. So we get the following Theorem.

Theorem 16. $\{a^n b^{n^k} \mid n \in \mathbf{N}\} (k = 1, 2, \dots)$ can be recognized by $(k+1)$ TFA in linear time.

Proof. The idea to prove this theorem is almost the same as the one in the previous theorem. Let \mathcal{T}_1 scan symbol b , and the numbers of symbols a having been scanned by $\mathcal{T}_i (i = 2, 3, \dots, k+1)$ are acting like a k digits of n -radix number. The details are similar to the above theorem, and thus we do not bore here. □

Corollary 17. $\{a^n b^{rn^{k-1}} \mid n \in \mathbf{N}, r \in \mathbf{N}\} (k = 1, 2, \dots)$ can be recognized by k TFA in linear time.

Proof. Let n_k be the number of symbol a having been scanned by \mathcal{T}_k . When n_k reaches n , and if there are symbols b in \mathcal{T}_1 which have not been scanned, we do not reject the input, but return tape head of \mathcal{T}_k to \mathcal{C}_k and go on. The other details are the same as k TFA recognizing language $\{a^n b^{n^{k-1}} \mid n \in \mathbf{N}\}$. □

Theorem 18. For any $\epsilon > 0$, there is a k TQCFA \mathcal{M} that accepts any $\omega \in \{a^n b^{n^k} \mid n \in \mathbf{N}\}$ with certainly, rejects any $\omega \notin \{a^n b^{n^k} \mid n \in \mathbf{N}\}$ with probability at least $1 - \epsilon$ and halts in polynomial time.

Proof. In terms of Colollary 17, we can check whether w is of the form $a^n b^{rn^{k-1}}$ in linear time. Then we use the similar method as in Theorem 12 to verify $r = n$. The process is

similar to that in Theorem 12, except the random walk. k TQCFA will take scanning n^{k-1} symbols b as one step in the random walk. We omit the details here. \square

6 Conclusions

In this paper, we have studied 2TFA and proved several languages can be recognized by 2TFA. Augmenting with a quantum component of constant size, we have proposed a new computing model called 2TQCFA, and we have proved that $L_{square} = \{a^n b^{n^2} \mid n \in \mathbf{N}\}$ can be recognized by 2TQCFA in polynomial time with one-sided error. Furthermore, we have proposed m TFA and $MTQCFA$ and proved that $\{a^n b^{n^k} \mid n \in \mathbf{N}\}$ can be recognized by k TQCFA.

As we know, 2QCFA are more powerful than 2DFA. Thus, in the future we would like to consider this question: Are 2TQCFA more powerful than 2TFA? Furthermore, are k TQCFA more powerful than k TFA? We have proved that all languages which have been shown to be recognized by 2QCFA or 2QFA can also be recognized by 2TFA in this paper. It is natural to ask whether or not there is any language which can be recognized by 2QCFA or 2QFA but can not be recognized by 2TFA?

Acknowledgments

This work is supported in part by the National Natural Science Foundation (Nos. 60873055, 61073054), the Natural Science Foundation of Guangdong Province of China (No. 1025102750100004), the Fundamental Research Funds for the Central Universities (Nos. 10lgzd12,11lgpy36), the Research Foundation for the Doctoral Program of Higher School of Ministry of Education (Nos. 20100171110042, 20100171120051), the China Postdoctoral Science Foundation project (Nos. 20090460808, 201003375), and the project of SQIG at IT, funded by FCT and EU FEDER projects Quantlog POCI/MAT/55796/2004 and QSec PTDC/EIA/67661/2006, IT Project QuantTel, NoE Euro-NF, and the SQIG LAP initiative.

References

- [1] A. Ambainis, J. Watrous, Two-way finite automata with quantum and classical states, *Theoretical Computer Science* 287 (2002) 299-311.
- [2] A. Ambainis, M. Beaudry, M. Golovkins, A. Kikusts, M. Mercer, and D. Thénrien, Algebraic Results on Quantum Automata, *Theory of Computing Systems* 39 (2006), 1654-188.

- [3] A. Ambainis, R. Freivalds, One-way quantum finite automata: strengths, weaknesses and generalizations, in: Proceedings of the 39th Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, Palo Alto, California, USA, 1998, pp. 332-341. Also quant-ph/9802062, 1998.
- [4] A. Ambainis, A. Nayak, A. Ta-Shma, U. Vazirani, Dense quantum coding and quantum automata, *Journal of the ACM* 49 (4) (2002) 496-511.
- [5] A. Brodsky, N. Pippenger, Characterizations of 1-way quantum finite automata, *SIAM Journal on Computing* 31 (2002) 1456-1478. Also quant-ph/9903014, 1999.
- [6] L.Z. Li, D.W. Qiu, Determining the equivalence for one-way quantum finite automata, *Theoretical Computer Science* 403 (2008) 42-51.
- [7] L.Z. Li, D.W. Qiu, A note on quantum sequential machines, *Theoretical Computer Science* 410 (2009) 2529-2535.
- [8] D.W. Qiu, Characterization of Sequential Quantum Machines, *International Journal of Theoretical Physics* 41 (2002) 811-822.
- [9] D.W. Qiu, S. Yu, Hierarchy and equivalence of multi-letter quantum finite automata, *Theoretical Computer Science* 410 (2009) 3006-3017.
- [10] A. Kondacs, J. Watrous, On the power of finite state automata, in: Proceedings of the 38th IEEE Annual Symposium on Foundations of Computer Science, 1997, pp. 66-75.
- [11] C. Moore and J.P. Crutchfield, Quantum automata and quantum grammars, *Theoretical Computer Science* 237 (2000) 275-306. Also quant-ph/9707031, 1997.
- [12] D.W. Qiu, Some Observations on Two-Way Finite Automata with Quantum and Classical States, ICIC 2008, LNCS 5226, pp.1-8, 2008.
- [13] D.W. Qiu, L.Z. Li, An overview of quantum computation models: quantum automata, *Frontiers of Computer Science in China* 2 (2)(2008) 193-207.
- [14] L.K. Grover, A fast quantum mechanical algorithm for database search, in: Proceedings of the 28th Annual ACM Symposium on Theory of Computing, Philadelphia, Pennsylvania, USA, 1996, pp. 212-219.
- [15] J.E. Hopcroft, J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, New York, 1979.
- [16] J. Gruska, *Quantum Computing*, McGraw-Hill, London, 1999.
- [17] J. Preskill, Quantum information and computation, Lecture Notes for Physics, Vol. 229, California Institute of Technology, 1998./ Theory of Quantum Information, Lecture Notes, University of Waterloo.

- [18] M.A. Nielsen, I.L. Chuang, Quantum Computation and Quantum Information, Cambridge University Press, Cambridge, 2000.
- [19] M.Siper, Introduction to the theory of computation, PWS, Boston, 1997.
- [20] M. V. Panduranga Rao, Interference automata, Theoretical Computer Science 403 (2008) 89-103.
- [21] P.W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM Journal on Computing 26 (5) (1997) 1484-1509.
- [22] Rain. M. and Scott. D. ,Finite automata and their decision problems,IBM J. Res. Develop. 3 (1959) 114-125.
- [23] S.Yu, Regular Languages, in:Handbbok of Formal Languages(G.Rozenberg, A.Salomaa, eds.), Spring-Verlag, 1998, pp.41-110.
- [24] W. Feller, An Introduction to Probability Theory and its Applications, Vol. I, Wiley, New York, 1967.